NL 010110

US

J1011 U.S. PTO
10/073772
02/11/02

Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

01200505.4

CERTIFIED COPY OF PRIORITY DOCUMENT

Der Präsident des Europäischen Patentamts; Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets p.o.

[illegible]

I.L.C. HATTEN-HECKMAN

DEN HAAG, DEN
THE HAGUE, 15/10/01
LA HAYE, LE

EPA/EPO/OEB Form 1014 - 02.91

Europäisches Patentamt | European Patent Office | Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.: Application no.: Demande n°:	01200505.4	Anmeldetag: Date of filing: Date de dépôt:	12/02/01

Anmelder:
Applicant(s):
Demandeur(s):
Koninklijke Philips Electronics N.V.
5621 BA Eindhoven
NETHERLANDS

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Robust hashing of multimedia content

In Anspruch genommene Prioriät(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

Robust hashing of multimedia content

FIELD OF THE INVENTION

The invention relates to a method and arrangement for generating a hash signal identifying an information signal. The invention also relates to a method and arrangement for matching such a hash signal with hash signals stored in a database.

BACKGROUND OF THE INVENTION

Hash functions are generally known in the field of cryptography, where they are amongst others used to summarize large amounts of data. For instance in order to verify that a large file has been received, it suffices to send the hash value (also referred to as signature) of that file. If the returned hash value matches the hash value of the original file there is almost complete certainty that the file is correctly received by the receiving party. The remaining uncertainty is introduced due to the fact that a *collision* might occur: i.e. two different files may have the same hash value. A carefully designed hash function should minimize the probability of collision.

A particular property of a cryptographic hash is its extreme fragility. Flipping a single bit in the source data will in general result in a completely different hash value. This makes cryptographic hashing unsuitable for summarizing multimedia content where different quality versions of the same content should yield the same summary. Summaries of multimedia content that are to a certain extent invariant to data processing (as long as the processing retains an acceptable quality of the content) are referred to as a *robust* summaries or, which is our preferred naming convention, *robust hashes*. Using a database of robust hashes and content identifiers, unknown content can be identified, even it is degraded (e.g. by compression or AD/DA conversion).

Using a robust hash to identify multimedia content is an alternative for using watermarking technology for the same purpose. There is however also a great difference. Whereas watermarking requires action on original content (viz. watermark embedding) before being released, with its potential impact on content quality and logistical problems, robust hashing requires no action before release. The downside of hashing technology that access to a database is needed (e.g. hashing is only viable in a connected context), whereas watermark detectors can operate locally (for example in non-connected DVD players).

United States Patent 4,677,466 discloses a known method of deriving a signature from a television signal for broadcast monitoring. In this prior art method, the signature is derived from a short video or audio sequence after the occurrence of a specified event such as a blank frame.

OBJECT AND SUMMARY OF THE INVENTION

It is a general object of the invention to provide a robust audio hashing technology. More particularly, it is a first object of the invention to provide a method and arrangement for extracting a limited number of hashing bits from multimedia content. The hashing bits are robust, but not in a sense that the probability of bit errors is zero. It is known that non-exact pattern matching (i.e. searching for the most similar hash value in the database) is NP-complete. In layman terms, this means that the best search strategy is exhaustive search, which is prohibitive in many applications dealing with large databases. Therefore, a second object of the invention is to provide a method and arrangement that overcomes this NP-complete search complexity.

The first object is achieved by by dividing the information signal in successive (preferably overlapping) time intervals, computing a hash value for each time interval, and concatenating successive hash values to constitute the hash signal. The hash value is computed by thresholding a scalar property or a vector of properties of the information signal, for example, the energy of disjoint frequency bands.

The second object is achieved by selecting a single hash value of said input block of hash values, searching said hash value in the database, calculating a difference between the input block of hash values and a corresponding stored block of hash values. These steps are repeated for further selected hash values until said difference is lower than a predetermined thershold.

Further features of the invention are defined in the subclaims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a schematic diagram of an embodiment of an arrangement for generating a hash signal in accordance with the invention.

Fig. 2 shows a diagram illustrating the subdivision of the multimedia signal spectrum in logarithmic spaced bands.

Figs. 3A-3C show diagrams illustrating hash values extracted from an audio clip.

Fig. 4 shows a flow diagram of operations carried out by a computer which is shown in Fig. 1.

Fig. 5 shows a further diagram to illustrate the operation of the computer which is shown in Fig. 1.

Fig. 6 shows a graph of the number or bit errors in 256 hash values forming an extracted hash block which is shown in Fig. 3B.

Fig. 7 shows a graph of the most reliable bit of the 256 hash values of the hash block which is shown in Fig. 3B.

Fig. 8 shows a flow diagram of operations carried out by the computer which is shown in Fig. 1 in accordance with a further embodiment of the invention.

DESCRIPTION OF EMBODIMENTS

Hereinafter, robust hashing of audio signals will be described, but the invention can also be applied to speech, video and image signals. We will assume that we are dealing with mono audio that has been sampled with a sample frequency of 44.1 kHz (CD-quality). If the audio is stereo there are two options: either we extract hash values for the left and the right channel separately or we add the left and the right channel and then extract hash values.

Before describing a preferred embodiment, a general description of considerations underlying this invention will be elucidated.

Two audio signals can differ quite drastically (e.g. by compression) in a signal theoretic sense, whereas the human auditory system (HAS) has no problem in recognizing the 'sameness'. Ideally, a hash function mimics the behavior of the human perceptual system (HPS), i.e. produces the same hash signal for content that is considered the same by the HPS. However, many kinds of processing (compression, noise addition, echo addition, D/A and A/D conversion, equalization etc.) can be applied to the audio and there is no algorithm that is able to mimic the HPS perfectly. A complicating factor is that even the HAS varies from person to person as well as in time, and even the notion of one single HAS is untenable. Also, the classical definition of a hash does not take time into account: a robust hash should not only be able to identify the content, but should also be able to identify time (intervals). For this reason the following definition for a robust hash is herein used:

A robust hash is a function that associates to every basic time-unit of audio content a semi-unique bit-sequence that is continuous with respect to content similarity as perceived by the HPS.

In other words, if the HPS identifies two pieces of audio as being very similar, the associated hash values should also be very similar. In particular, if we compute the hash values for original content and compressed content, the hash values should be similar. Also, if hash values are computed for overlapping time intervals, the hash values should be similar, i.e. hash values should have a low pass character. On the other hand, if two signals really represent different content, the robust hash should be able to distinguish the two signals (semi-unique). This is similar to the collision requirement for classical cryptographic hashes. The required robustness of the hashing function is achieved by deriving the hash function from robust features (properties), i.e. features that are to a large degree invariant to processing. Robustness can be expressed by the Bit Error Rate (BER), which is defined as the ratio of the number of erroneous bits and the total number of bits.

If we only have a short piece of audio (in the order of seconds), we also like to determine which song it is. As audio can be seen as an endless stream of audio-samples, it is necessary to subdivide audio signals into time intervals or *frames* and to calculate a hash value for every frame.

Very often, when trying to match hash values in a database, it is impossible to determine the frame boundaries. This synchronization problem is particularly applicable to audio hashing. It is solved by dividing the signal in overlapping frames. Overlapping also assures that hash values of contiguous frames have a certain amount of *correlation*. In other words, the hash values change slowly over time.

Fig. 1 shows a schematic diagram of an embodiment of an arrangement for generating the hash signal in accordance with the invention. The audio signal is first downsampled in a downsampler 11 to reduce the complexity of subsequent operations and restrict the operation to a frequency range of 300-3000 Hz, which is most relevant for the Human Auditory System.

In a framing circuit 12, the audio signal is divided in frames. The frames are weighed by a Hanning window, have a length of 16384 samples (≈0.4 seconds) and an overlap factor of 31/32. The overlap is chosen in such a way to assure a high correlation of the hash values between subsequent frames. The spectral representation of every frame is computed by a Fourier transform circuit 13. In the next block 14, the absolute value of the (complex) Fourier coefficients is computed.

A band division stage 15 divides the spectrum in a number (e.g. 33) of bands. In Fig. 1, this is schematically shown by selectors 151, each of which selects the Fourier coefficients of the respective band. In a preferred embodiment of the arrangement, the bands have a logarithmic spacing, because the HAS also operates on approximately logarithmic bands. By choosing the bands in this manner, the hash value will be less susceptible to processing changes such as compression and filtering. In the preferred embodiment the first band starts at 300Hz and every band has a bandwidth of one musical tone (i.e. the bandwidth increases by a factor of $2^{1/12}$≈1.06 per band). Fig. 2 shows an example of the subdivision of the spectrum in logarithmic spaced bands.

Next, for every band a certain (not necessarily scalar) characteristic property is calculated. Examples of properties are energy, tonality and standard deviation of the power spectral density. In general the chosen property can be an arbitrary function of the Fourier coefficients. Experimentally it has been verified that the energy of every band is a property that is most robust to many kinds of processing. This energy computation is carried out in an energy computing stage **16**. For each band, it comprises a stage **161** which computes the sum of the absolute values of the Fourier coefficients in the band.

In order to get a binary hash value, the robust properties are subsequently converted into bits. The bits can be assigned by calculating an arbitrary function of the robust properties of possibly different frames and then comparing it to a threshold value. The threshold itself might also be a result of another function of the robust property values.

In the present arrangement, a bit derivation circuit **17** converts the energy levels of the bands into a binary hash value. In a simple embodiment, the bit derivation stage generates one bit for each band, for example, a '1' if the energy level is above a threshold and a '0' if the energy level is below said threshold. The thresholds may vary from band to band. Alternatively, a band is assigned a hash value bit '1' if its energy level is larger than the energy level of its neighbor, otherwise the hash value bit is '0'. The present embodiment uses an even improved version of the latter alternative. To avoid that a major single frequency in the audio signal would produce identical hash values for successive frames, variations of the amplitude over time are also taken into account. More particularly, a band is assigned a hash value bit '1' if its energy level is larger than the energy level of its neighbor and if that was also the case in the previous frame, otherwise the hash value bit is '0'. If we denote the energy of a band *m* of frame *n* by *EB(n,m)* and the *m*-th bit of the hash value *H* of frame *n* by *H(n,m)*, the bit derivation circuit **17** generates the bits of the hash value in the following manner:

$$H(n,m)=\begin{cases}1 \text{ if } EB(n,m)-EB(n,m+1)-(EB(n-1,m)-EB(n-1,m+1))>0\\0 \text{ if } EB(n,m)-EB(n,m+1)-(EB(n-1,m)-EB(n-1,m+1))\leq 0\end{cases}$$

To this end, the bit derivation circuit **17** comprises, for each band, a first subtractor **171**, a frame delay **172**, a second subtractor **173**, and a comparator **174**. The 33 energy levels of the spectrum of an audio frame are thus converted into a 32-bits hash value. The hash values of successive frames are finally stored in a buffer **18**, which is accessible by a computer **20**. The computer stores the robust hashes of a large number of original songs in a database **21**.

In a subsequent operation, the same arrangement computes the hash values of a unknown audio clip. Fig. 3A shows the hash values of 256 successive overlapping audio frames (≈3 seconds) of the audio clip as stored in the database **21**. In the Figure, each row is a 32-bit hash value, a white pixel represents a '1' bit of the hash value, a black pixel represents a '0' bit, and time proceeds from top to bottom. Fig. 3B shows the hash values extracted from the received audio clip after MP3 compression at 32 kBit/s. Ideally these two figures should be identical, but due to the compression some bits are different. The difference is shown in Fig. 3C.

The process of matching the extracted hash values to the hash values in a large database will now be described. This is a non-trivial task since it is well-known that imperfect matching (remember that the extracted hash values may have bit errors) is NP-complete. We will show this with the following example, which is using the preferred embodiment described above. In this example database we take 100,000 songs of approximately five minutes (≈25000 hash values per song). We will assume that we extracted 256 hash values (the hash block which is shown in Fig. 3B) from the unknown audio clip, and have to determine to which of the 100,000 stored songs the extracted hash block matches best. Hence the position of a hash block in one of the 100,000 songs has to be found, which most resembles the extracted has block, i.e. for which the bit error rate (BER) is minimal or, alternatively, for which the BER is lower than a certain threshold. The threshold value directly determines the false positive rate, i.e. the rate at which songs are incorrectly identified from the database.

In order to determine this threshold, the distribution of the bit errors for songs not present in the database has to be analyzed.. If the hash extraction process yields random, independent and identically distributed hash bits, the bit errors will have a binomial distribution *(n,p)*, where *n* equals the amount of bits extracted and *p* is the probability that a bit is wrong. Since *n* (=8192=32×256) is large in our application the binomial distribution can be approximated by a normal distribution with a mean *np* and standard deviation $\sqrt{(np(1-p))}$. However the robust hash values have high correlation on the time axis. This correlation is due to the correlation that is inherently present in audio, but is also introduced by the overlap in the framing that is used during hash extraction. Experiments have shown that the bit errors of the robust hash values of the preferred embodiment have a normal distribution, the standard deviation being around 3 times the standard deviation compared to the standard deviation if the bits of the hash values were independent and identically distributed.

The threshold for the BER used during experiments was 0.25. This means that of 8192 bits less than 2048 bit errors have to occur in order to decide that the hash values originate from the same song. The bit errors have in this case a normal distribution with a mean μ of $np=4096$ and a standard deviation σ of $3\sqrt{(np(1-p))}=135.76$. The chosen threshold setting then corresponds with the false alarm probability of 15.2σ. Hence the false alarm probability equals $1.8\cdot10^{-52}$. Note however that in practice the false alarm probability will be higher if music with similar hash values (e.g. a piece of Mozart played by two different pianists) are included in the database.

Searching the position of the extracted hash block in the database can be done by brute force matching. This will take around 2.5 billion (=25000×100,000) matches. Moreover the number of matches increases linearly with the size of the database.

In accordance with an aspect of the invention, the computer **20** uses a more efficient strategy for finding the corresponding song in the database **21**. Fig. 4 shows a flow diagram of operations carried out by the computer. Upon storing an original song in the database, the computer updates a lookup table (LUT) in a step **40**. The LUT is shown as a separate memory **22** in Fig. 1 but it will be appreciated that it will part of the large database memory **21** in practice. As shown in Fig. 5, the LUT **22** has an entry for each possible 32-bit hash value. Each entry of the LUT points to the song(s) and the position(s) in that song where the respective hash value occurs. Since a hash value can occur at multiple positions in multiple songs the song pointers are stored in a linked list. Thus the LUT can generate multiple candidate songs. Note that a LUT containing 2^{32} entries can be impractical when there are only a limited number of songs in the database. In such a case it is advantageous to implement the LUT with a hash table and a linked list. Reference numeral **50** in Fig. 5 denotes the block of 256 hash values extracted from the unknown audio clip (the hash block which is shown in Fig. 3B).

In a first embodiment of the matching method, it will be assumed that every now and then a single hash value has no bit errors. In a step **41**, a single hash value H(m) is selected from the hash block and sent to the database. Initially, this will be the last hash value H(256) of the extracted hash block. In the example shown in Fig. 5, this is the hash value 0x00000001. The LUT in the database is pointing to a certain position in song 1. Lets say that this position is position *p*. In a step 42, the computer calculates the BER between the extracted hash block and the block of hash values from position *p-255* until position *p* of song 1 (denoted 51 in Fig. 5). In a step 43, it is checked whether the BER is low (<0.25) or high. If the BER is low, the probability is high that the extracted hash values originate from song 1. If the BER is high, either the song is not in the database or the single hash value H(m) contains an error. The latter will be assumed to be the case in this example. Another single hash value is then selected in a step 44 and looked up in the LUT (step 41). In Fig. 5, the last but one single hash value H(255) is now being looked up. This hash value appears to occur in song 2. The BER between input block 50 and stored block 52 appears to be lower than 0.25 now, so that song 2 is identified as the song from which the audio clip originates. Note that the last hash value in the stored block 52 is 0x00000000. Apparently, the previously selected hash value 0x0000001 had one bit error.

The computer thus only looks at one single hash value at a time and assume that every now and then such a single hash value has no bit errors. The BER of the extracted hash block is then compared with the (on the time axis) corresponding hash blocks of the candidate songs. The title of the candidate song with the lowest BER will be chosen as the song from where the extracted hash values originate, provided that the lowest BER is below the threshold (step 45). Otherwise the database will report that the extracted hash block was not found. Another single hash value will then be tried. If none of the single hash values leads to success (step 46), the database will respond by reporting the absence of the candidate song in the database (step 47).

The above described method relies on the assumption that every now and then an extracted hash value has no bit errors, i.e. is perfectly equal to the corresponding stored hash value. Extensive experiments have shown that this occurs regularly a few times per second for most audio. This is for example shown in Fig. 6, which shows the number or bit errors in the 256 hash values forming the extracted block of Fig. 3B. Thirteen hash values occur without any bit errors in this 3 seconds audio clip.

However, it is unlikely that hash values without any bit errors occur when the audio is severely processed. In that case we will be unable to retrieve the title of the song with the previous method. To this end, an embodiment of the matching method uses soft information of the hash extraction algorithm to find the extracted hash values in the database. By soft information we mean the reliability of a bit, or the probability that a hash bit has been retrieved correctly. In this embodiment, the arrangement for extracting the hash values includes a bit reliability determining circuit 19 (see Fig. 1). This circuit receives the differential energy band levels in the form of real numbers. If that real number is very close to the threshold (which is zero in this example), the respective hash bit is unreliable. If instead the number is very far from the threshold, it is a reliable hash bit. The bit reliability determining circuit 19 derives the reliability of every hash bit, and thus enables the computer 20 to generate a list of most probable alternative hash values for each hash value. By assuming again that at least one of the alternative hash values is correct, the song title can be received correctly and easily. Fig. 7 shows, for all the 256 hash values of the hash block of Fig. 3B, which bit of the hash value the most reliable one is.

Fig. 8 shows a flow diagram of operations carried out by the computer in this embodiment of the method of finding the extracted hash values in the database. The same reference numerals are used for operations already described before. Again, the last extracted hash value (0x00000001, see Fig. 5) of the hash block is initially selected and sent to the database (step 41). The LUT in the database is pointing to position *p* in song 1. The BER between the extracted hash block and the corresponding block (from position *p-255* until position *p*) in song 1 is calculated (step 42). From the earlier example, we meanwhile know that the BER is high because the applied hash value contains an error. In a step 81, the computer now consults the bit reliability determining circuit 19 (Fig. 1) and learns that bit 0 is the least reliable bit of this particular hash value. The next most probable candidate hash value is now obtained by flipping said bit. The new hash value (0x00000000) is sent to the database in a step 82. As shown in Fig. 5, the hash value 0x00000000 leads to two possible candidate songs in the database: song 1 and song 2. If now for example the extracted hash values have a low BER with the hash values of song 2, song 2 will be identified as the song from which the extracted hash values originate. Otherwise new hash value candidates will be generated, or another hash value will be used to try to find the respective song in the database. This strategy is continued until it is found in a step 83 that there are no further alternative candidate hash values.

Note that in practice once a piece of audio is identified as originating from a certain song, the database can first try to match the extracted hash values with that song before generating all the candidate hash values.

A very simple way of generating a list of most probable hash values is to include all the hash values with *N* most reliable bits fixed and for the remaining bits every possible combination. In the case of 32 bits per hash and choosing *N=23* implies a list of 512 candidate hash values. Furthermore it implies that the 9 least reliable bits of the hash value can be wrong before an audio excerpt cannot be identified anymore. For the case shown in Figure 6 this means that 117 hash values, instead of 13 with the previous method, will yield a correct pointer to the song in the database.

A few applications of robust hashing will now be described.

- Broadcast Monitoring: A broadcast monitoring system consists of two parts: a central database containing the hash values of a large number of songs and monitoring stations that extract hash values for the audio that is broadcast by for instance radio stations. The monitoring station will send the extracted hash values to the central database and then the database will be able to determine which song has been broadcast.
- Mobile Phone Audio Info: Imagine that you are in a bar and hear a song of which you want to know the title. You then just pick up your mobile telephone and phone up an audiohash database. The audiohash database will then hear the song and extract hash values. If it then finds the hash values of the song in the database it will report back the title.
- Connected Content (MediaBridge): The company Digimarc has at the moment an application they call MediaBridge, which is based on watermarking technology. The idea is that a watermark in a piece of multimedia will direct a user to a certain URL on the Internet where he can get some extra information. E.g. an advertisement in a magazine is watermarked. By holding this advertisement in front of a webcam, a watermark detector will extract a watermark key that is send to a database. This database then contains the URL to which the user will be redirected. The same application can work with the use of robust hashing technology. In the future one might even think of a person pointing his mobile videophone at a real life object. The audio hash database will then report back information about this object, either directly or via an URL on the Internet.
- Multimedia Quality Metering: If the hash values of high quality original content are listed in the database a quality measure can be obtained by determining the BER of the extracted hash values of processed multimedia content.

From an abstract viewpoint, the robust audio hashes are derived from an audio signal by comparing energy in different frequency bands and over time. A generalization of this approach is to consider any cascade of LTI and non-linear functions. In particular, a robust hash can also be obtained by applying a (dyadic) filter bank (an LTI operator), followed by squaring of taking absolute values (a non-linear function), followed by a difference operator over time and/or band (an LTI operator), finally followed by a thresholding operator. By applying a carefully designed linear filter bank as an initial operator, the complexity of a FFT can be avoided. Moreover, as many compression engines have a linear filter bank as an initial phase, there is the option to integrate feature extraction with compression.

It is further noted that robust hashing and digital watermarks can be used in combination to identify content. The method described above and some watermark detection algorithms have a number of initial processing steps in common, viz. the computation of the spectral representation. This leads to the idea that watermark detection and feature extraction can easily be integrated in one application. Both retrieved watermark and hash values can then be sent to a central database for further analysis, to allow identification of content.

In summary, the disclosed method generates *robust* hashes for multimedia content, for example, audio clips. The audio clip is divided (12) in successive (preferably overlapping) time intervals. For each time interval, the frequency spectrum is divided (15) in bands. A robust property of each band (e.g. energy) is computed (16) and represented (17) by a respective hash bit. An audio clip is thus represented by a concatenation of binary hash values, one for each time interval. To identify a possibly compressed audio signal, a block of hash values derived therefrom is matched by a computer (20) with a large database (21). Such matching strategies are also disclosed. In an advantageous embodiment, the extraction process also provides information (19) as to which of the hash bits are the least reliable. Flipping these bits considerably improves the speed and performance of the matching process.

CLAIMS:

1. A method of generating a hash signal identifying an information signal, comprising the steps of:
- dividing the information signal in successive time intervals,
- computing a hash value for each time interval, and
- concatenating successive hash values to constitute the hash signal.

2. The method of claim 1, wherein said time intervals are overlapping.

3. The method of claim 1, wherein said computing step comprises computing the hash value by thresholding a scalar property or vector of properties of the information signal within the time interval.

4. The method of claim 3, further comprising the step of using the input of said threholding step to generate information indicative of the reliablity of the bits of the hash value.

5. The method of claim 3, wherein said computing step comprises the steps of:
- transforming the information signal within the time interval into disjoint bands;
- calculating a property of the signal in each of said bands;
- comparing the properties in the bands with respective thresholds;
- representing the results of said comparisons by respective bits of the hash (sample) value.

6. The method of claim 5, wherein the bands are frequency bands having an increasing bandwidth as function of the frequency.

7. The method of claim 5, wherein said property is the energy of a band.

8. The method of claim 5, wherein said property is the tonality of a band.

9. A method of matching an input block of hash values representing at least a part of an information signal with hash signals identifying respective information signals stored in a database, comprising the steps of:

(a) selecting a hash value of said input block of hash values;

(b) searching said hash value in the database;

(c) calculating a difference between the input block of hash values and a stored block of hash values in which the hash value found in step (b) has the same position as the selected hash value in the input block;

(d) repeating steps (a)-(c) for a further selected hash value until said difference is lower than a predetermined thershold.

10. The method of claim 9, wherein the further selected hash value is another hash value of the input block of hash values.

11. The method of claim 9, wherein the further selected hash value is obtained by reversing a bit of the previously selected hash value.

12. The method of claim 11, further comprising the steps of receiving information indicative of the reliability of the bits of the selected hash value, and using said information to determine the bit to be reversed.

13. An arrangement for generating a hash signal identifying an information signal in accordance with a method as claimed in any one of claims 1-8.

14. An arrangement for matching an input block of hash values representing at least a part of an information signal with hash signals identifying respective information signals stored in a database in accordance with a method as claimed in any one of claims 9-12.

15. A method of redirecting a receiver of an information signal to an Internet website, comprising the steps of deriving from said information signal a hash signal, and matching said hash signal with hash signals identifying Internet websites stored in a database.

16. A method of measuring the quality of an information signal, comprising the steps of deriving from said information signal a hash signal, matching said hash signal with a hash

signal identifying said information signal stored in a database, and calculating the difference between the derived hash signal and the stored hash signal.

17. A method of identifying a multimedia signal, comprising the steps of receiving and/or recording at least a part of said multimedia signal, deriving from said multimedia signal a hash signal, sending said hash signal to a database for matching it with hash signals stored in said database, and receiving from said database an identifier of the multimedia signal.

18. The method of claim 17, wherein said steps of receiving and/or recording the multimedia signal, deriving and sending the hash signal, and receiving the identifier are performed by a mobile telephone device.

ABSTRACT:

Hashes are short summaries or signatures of data files which can be used to identify the file. Hashing multimedia content (audio, video, images) is difficult because the hash of original content and processed (e.g. compressed) content may differ significantly.

The disclosed method generates *robust* hashes for multimedia content, for example, audio clips. The audio clip is divided (12) in successive (preferably overlapping) time intervals. For each time interval, the frequency spectrum is divided (15) in bands. A robust property of each band (e.g. energy) is computed (16) and represented (17) by a respective hash bit. An audio clip is thus represented by a concatenation of binary hash values, one for each time interval. To identify a possibly compressed audio signal, a block of hash values derived therefrom is matched by a computer (20) with a large database (21). Such matching strategies are also disclosed. In an advantageous embodiment, the extraction process also provides information (19) as to which of the hash bits are the least reliable. Flipping these bits considerably improves the speed and performance of the matching process.

Fig. 1.

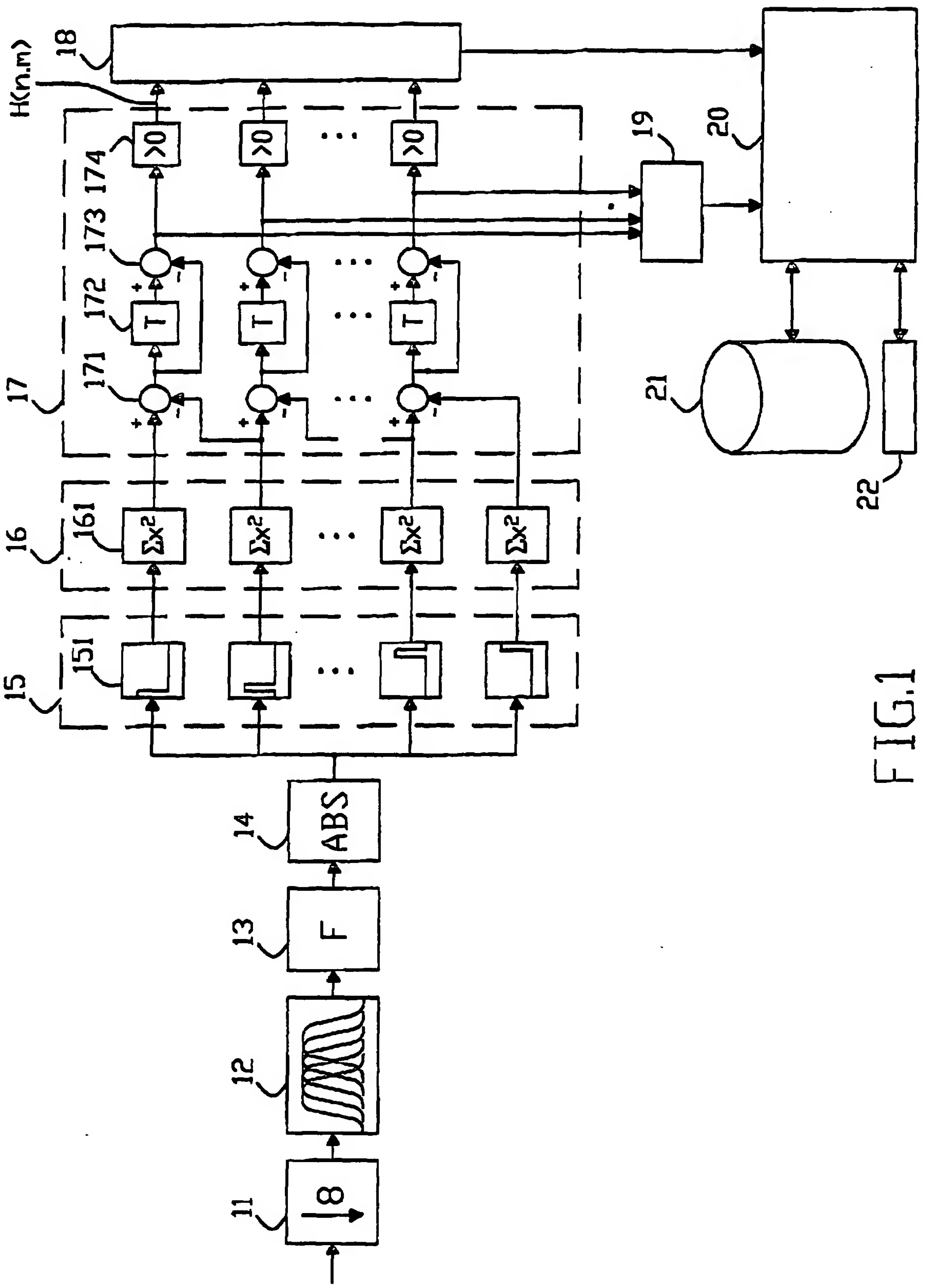
11
12
13
F
14
ABS
15
151
16
161
Σx²
17
171
172
T
173
174
>0
+
-
18
H(n,m)
19
20
21
22

FIG.1

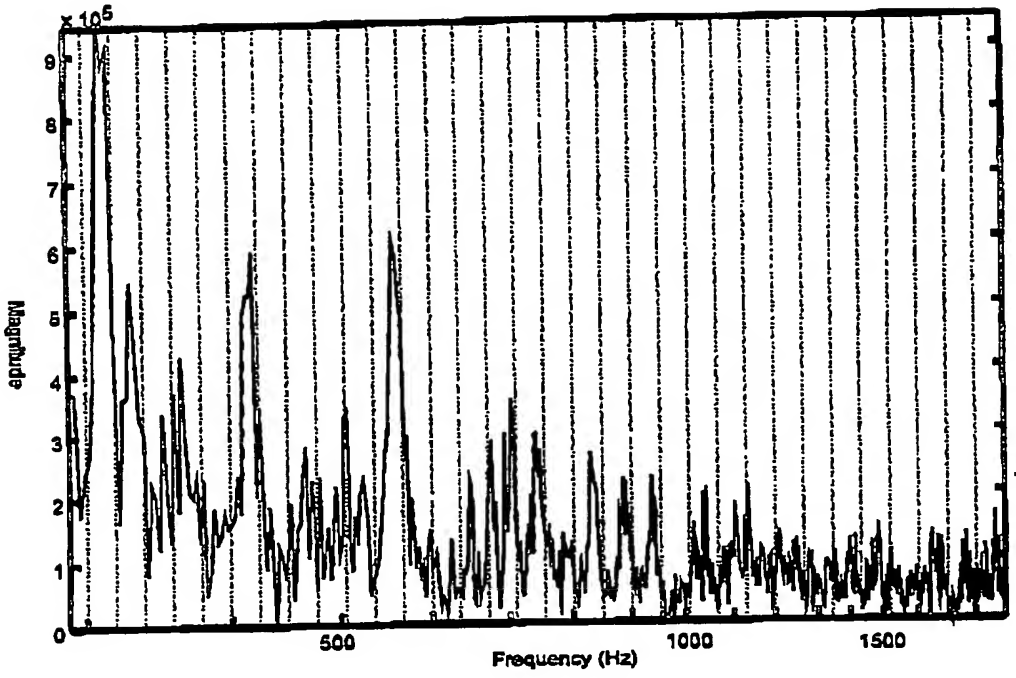
x 10^5
9
8
7
6
5
4
3
2
1
0
Magnitude
500
1000
1500
Frequency (Hz)

FIG.2

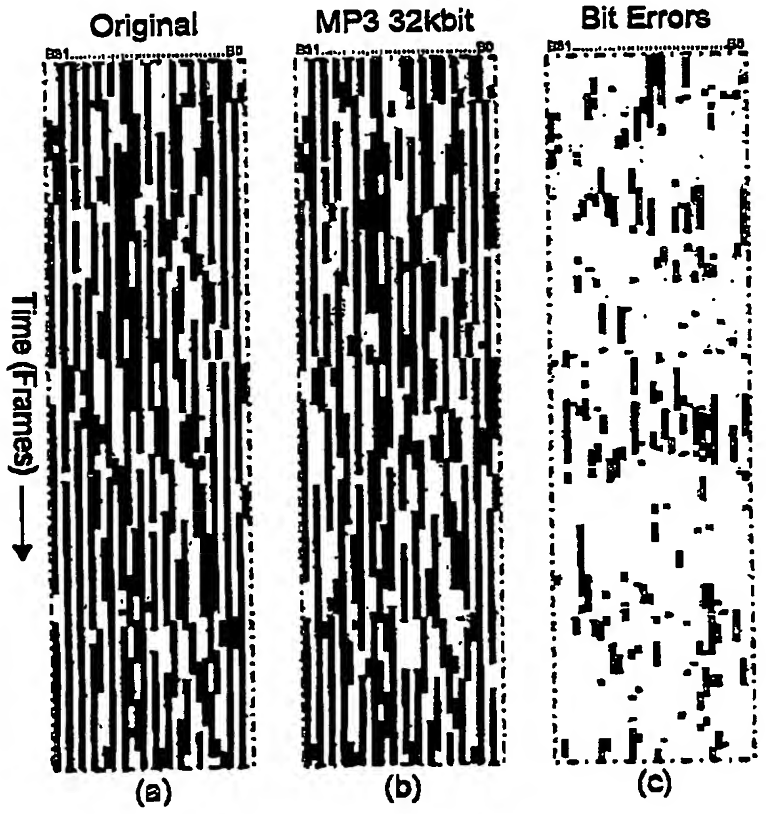
Original
MP3 32kbit
Bit Errors
B31
B0
Time (Frames)
(a)
(b)
(c)

FIG. 3

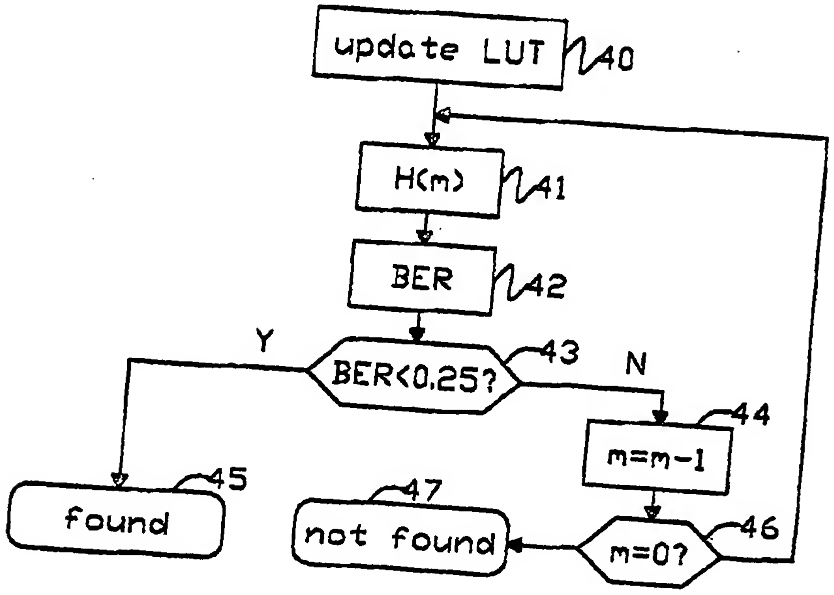
update LUT
40
H(m)
41
BER
42
Y
BER<0.25?
43
N
m=m-1
44
found
45
not found
47
m=0?
46

FIG.4

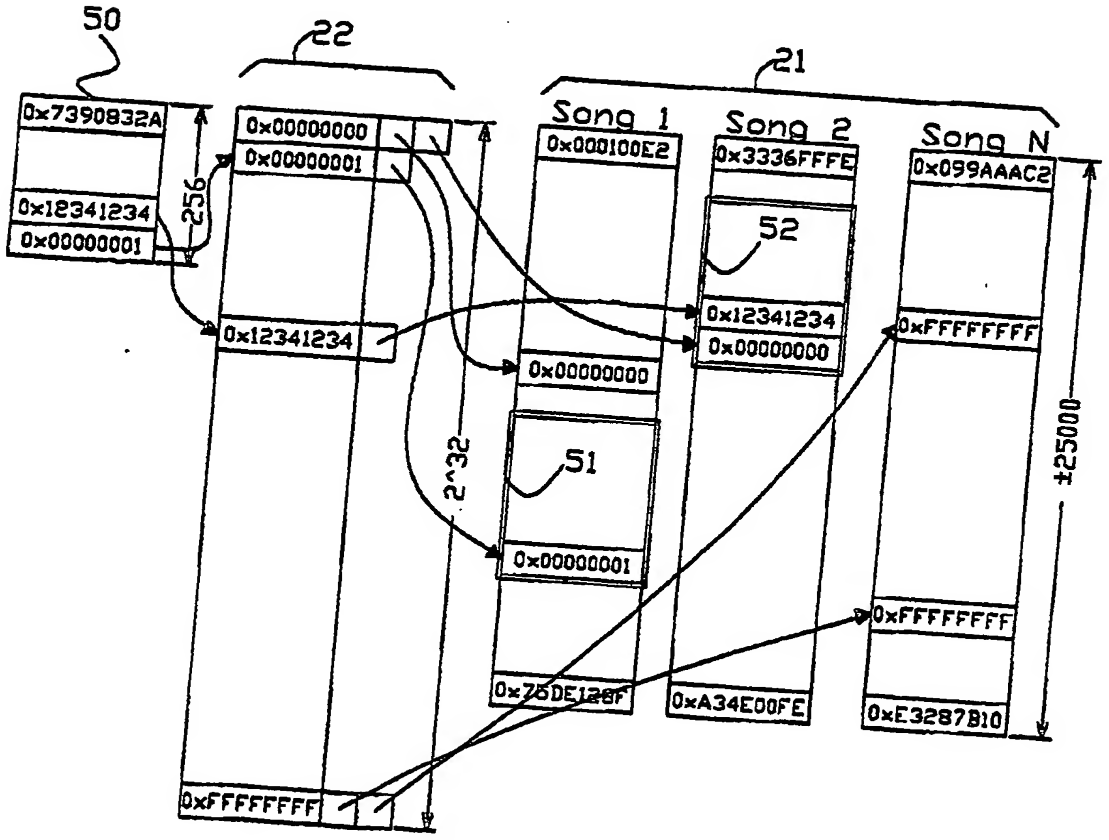
50
22
21
0x7390832A
0x12341234
0x00000001
256
0x00000000
0x00000001
0x12341234
2^32
0xFFFFFFFF
Song 1
0x000100E2
0x00000000
51
0x00000001
0x75DE128F
Song 2
0x3336FFFE
52
0x12341234
0x00000000
0xA34E00FE
Song N
0x099AAAC2
0xFFFFFFFF
0xFFFFFFFF
0xE3287B10
±25000

FIG.5

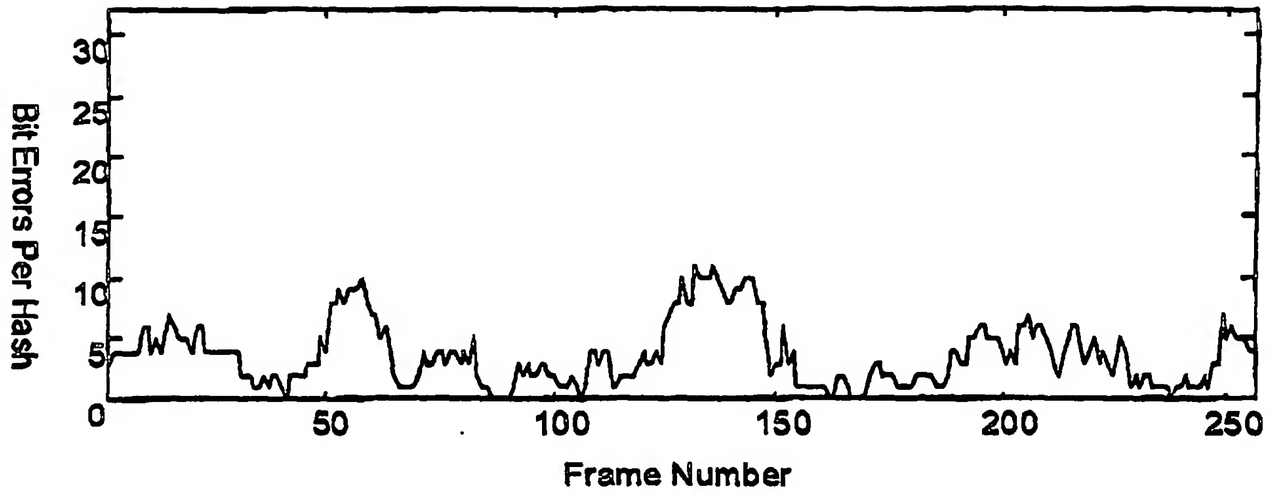
Bit Errors Per Hash
0
5
10
15
20
25
30
50
100
150
200
250
Frame Number

FIG. 6

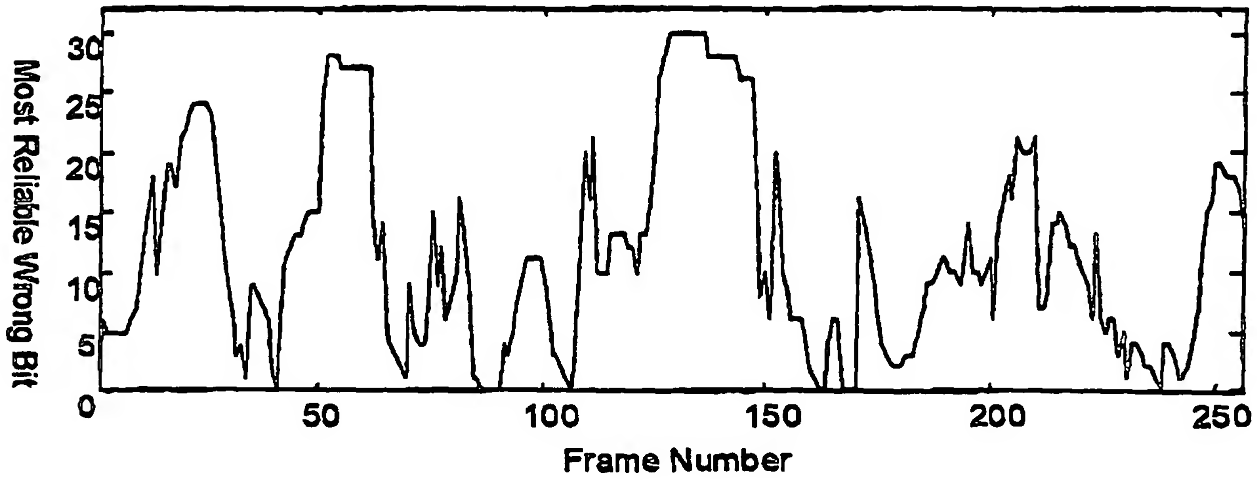
Most Reliable Wrong Bit
0
5
10
15
20
25
30
50
100
150
200
250
Frame Number

FIG. 7

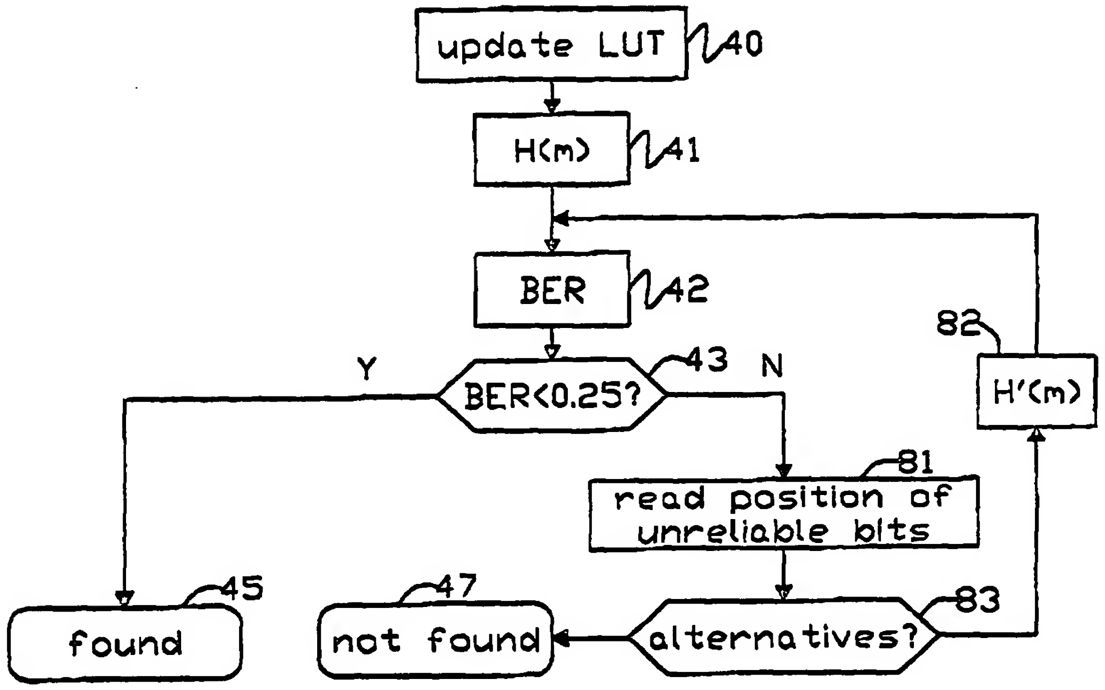
update LUT
40
H(m)
41
BER
42
BER<0.25?
43
Y
N
read position of unreliable bits
81
H'(m)
82
alternatives?
83
found
45
not found
47

FIG.8